

PLP - Primer Parcial - 2^{do} cuatrimestre de 2018

Este examen se aprueba obteniendo al menos dos ejercicios bien menos (B-) y uno regular (R), y se promociona con tres ejercicios bien menos (B-) y las respuestas adecuadas a las preguntas teóricas. Las notas para cada ejercicio son: -, I, R, B-, B. Poner nombre, apellido, número de orden y cantidad de hojas en la primera hoja, y numerar las hojas. Se puede utilizar todo lo definido en las prácticas y todo lo que se dio en clase, colocando referencias claras.

Ejercicio 1 - Programación funcional

En este ejercicio **no** se permite el uso de recursión explícita, a menos que se indique lo contrario. Pueden usar los ejercicios de la práctica o los vistos en clase, colocando referencias claras. Dar el **tipo** de todas las funciones definidas.

Este ejercicio consiste en implementar en Haskell varias funciones que permitan generar melodías y combinarlas.

Melodías

En el contexto de este trabajo, una melodía está compuesta por sonidos y silencios, dispuestos a lo largo del tiempo. Los sonidos se identifican por un tono, que corresponde a la “nota musical”, y se expresa mediante un número natural. Las duraciones se expresan también con un número natural, que corresponde a una cantidad de unidades de tiempo.

Se definirán los siguientes tipos en Haskell:

```
data Melodia = Silencio Duracion
              | Nota Tono Duracion
              | Secuencia Melodia Melodia
              | Paralelo [Melodia]

type Tono    = Integer
type Duracion = Integer
```

En el caso del constructor Secuencia, la segunda melodía comienza al instante siguiente a la finalización de la primera. En el caso de Paralelo, todas las melodías suenan simultáneamente.

Tanto los valores de los tonos como los de las duraciones serán siempre mayores o iguales que 0.

- Dar el tipo y definir la función `foldMelodia`, que implemente un esquema de recursión estructural (fold) para el tipo de datos `Melodia`. En este ejercicio, obviamente, se puede utilizar recursión explícita.
- Escribir la función `duracionTotal :: Melodia -> Duracion`, que dada una melodía devuelve la cantidad de instantes desde su comienzo hasta su fin.

Por ejemplo: `duracionTotal $`

`Paralelo [Nota 60 10, Secuencia (Silencio 3) (Nota 64 7), Secuencia (Silencio 6) (Nota 67 5)]`
devuelve 11.

- Definir la función `truncar :: Melodia -> Duracion -> Melodia`, que reproduce una melodía hasta una duración determinada, de manera que la duración total de la nueva melodía sea el mínimo entre la duración original y la indicada.

Ejercicio 2 - Cálculo Lambda Tipado

Se desea extender el Cálculo Lambda para soportar árboles con hojas distinguidas. Para esto introduciremos la siguiente sintaxis:

$$M ::= \dots \mid \text{Hoja}_{\sigma\tau}(M_1) \mid \text{Rama}(M_1, M_2) \mid \text{Bin}(M_1, M_2, M_3) \\ \mid \text{case } M_1 \text{ of } \text{Hoja}(h) \rightsquigarrow M_2; \text{Rama}(i, a) \rightsquigarrow M_3; \text{Bin}(a, i, b) \rightsquigarrow M_4 \\ (\sigma \text{ y } \tau \text{ representan el tipo de los nodos internos y el de las hojas respectivamente.})$$

La expresión $\text{Hoja}_{\sigma\tau}(M_1)$ representa un árbol de una única hoja con el valor de M_1 . $\text{Rama}(M_1, M_2)$ es un árbol con un nodo interno M_1 y un subárbol M_2 . $\text{Bin}(M_1, M_2, M_3)$ es un árbol con un nodo interno M_2 y dos subárboles M_1 y M_3 . También ampliaremos el conjunto de tipos de la siguiente manera: $\sigma ::= \dots \mid \text{AHD}_{\sigma\tau}$

a. Introducir las reglas de tipado para la extensión propuesta.

b. Exhibir una derivación del siguiente juicio de tipado:

$$\emptyset \triangleright \text{case Hoja}_{(\text{Bool} \rightarrow \text{Bool})\text{Nat}}(0) \text{ of Hoja}(x) \rightsquigarrow \text{isZero}(x); \text{Rama}(x, a) \rightsquigarrow \text{False}; \text{Bin}(a, x, b) \rightsquigarrow x \text{True} : \text{Bool}$$

c. Definir el conjunto de valores e introducir las reglas de semántica para esta extensión. No es necesario escribir las reglas de congruencia, basta con indicar cuántas son.

d. Mostrar paso por paso cómo reduce la expresión del punto b .

Ejercicio 3 - Inferencia de Tipos

En este ejercicio introduciremos al cálculo lambda tipado los árboles con información en las hojas (y sólo en ellas). Para esto introduciremos la siguiente sintaxis:

$$M ::= \dots \mid \text{Hoja}(M) \mid \text{Bin}(M_1, M_2) \mid \text{case } M_1 \text{ of Hoja}(h) \rightsquigarrow M_2; \text{Bin}(i, d) \rightsquigarrow M_3$$

También ampliaremos el conjunto de tipos de la siguiente manera: $\sigma ::= \dots \mid \text{AIH}_\sigma$

Se introducen las siguientes reglas de tipado para los nuevos términos:

$$\begin{array}{c} \frac{\Gamma \triangleright M : \sigma}{\Gamma \triangleright \text{Hoja}(M) : \text{AIH}_\sigma} \text{ (T-HOJA)} \quad \frac{\Gamma \triangleright M : \text{AIH}_\sigma \quad \Gamma \triangleright N : \text{AIH}_\sigma}{\Gamma \triangleright \text{Bin}(M, N) : \text{AIH}_\sigma} \text{ (T-BIN)} \\ \frac{\Gamma \triangleright M : \text{AIH}_\sigma \quad \Gamma \cup \{h : \sigma\} \triangleright N : \tau \quad \Gamma \cup \{i : \text{AIH}_\sigma, d : \text{AIH}_\sigma\} \triangleright O : \tau}{\Gamma \triangleright \text{case } M \text{ of Hoja}(h) \rightsquigarrow N; \text{Bin}(i, d) \rightsquigarrow O : \tau} \text{ (T-CASE)} \end{array}$$

a. Extender el algoritmo de inferencia para soportar la extensión propuesta.

b. Utilizar el algoritmo extendido para tipar la siguiente expresión, o mostrar que no tipa, indicando las sustituciones realizadas en cada paso.

$$\lambda x. \text{case Hoja}(x) \text{ of Hoja}(h) \rightsquigarrow \text{isZero}(h); \text{Bin}(i, d) \rightsquigarrow h$$

Preguntas Teóricas

Este ejercicio será corregido si el examen se encuentra en condición de promocionar. Entregar en una hoja separada al resto de los ejercicios.

a. Dada la siguiente función Haskell:

$$f \ a \ b = b : (a \ (-b))$$

Decir que representa la expresión $\text{fix } f \ 1$.

b. Mostrar un término $M \in \lambda^{\text{bn}}$ tal que $\emptyset \triangleright M : \text{Nat} \rightarrow \text{Nat}$ y

i) $\text{fix } M \Rightarrow V$ con V valor.

ii) no existe V tal que $\text{fix } M \Rightarrow V$ y V valor.

c. Mostrar, si es posible, dos términos $M_1 \neq M_2 \in \lambda^{\text{bn}}$ tal que:

■ $\emptyset \triangleright M_1 : \sigma, \emptyset \triangleright M_2 : \sigma'$, y $\text{Erase}(M_1) = \text{Erase}(M_2)$.

■ $\emptyset \triangleright M_1 : \sigma, \emptyset \triangleright M_2 : \sigma$, y $\text{Erase}(M_1) = \text{Erase}(M_2)$.